University of Beira Interior
Master Computer Engineering

Replication for Dependability on Virtualized Cloud Environments

Distributed Systems and Fault Tolerance

Tiago Freitas M6343
Claudio Rodrigues M6414

UBI
May of 2014

Goals

- Introduction
- Related Work
- Model
- Theoretical Analysis
- Experimental Evaluation
- Placement Strategies of the Defense
- Conclusion

Introduction

- To reduce costs, companies increasingly outsource information technology to cloud providers.
- Replication may not work well on the cloud.
- Common sense will tell us that one should not use a virtualized infrastructure to run replicas of the same service.
- Each new replica should run on a different PM.

Introduction (Cont)

- For economical reasons, clients will resort to the same cloud provider.
- Depending on the service, the defender may care for the number of PMs or the number of VMs that stand after each fault.
- Count the number of attacks that are necessary before the service loses the majority of machines, physical or virtual. (Byzantine algorithm)
- Two kinds of faults: crash-stop faults and intentional faults.

Related Work

- Replication is a well-know technique to tolerate faults in distributed computing systems.
- Replication can follow a primary-backup approach.
- The primary owns a centralized control of the group; or an active replication, also known as “state-machine approach” where all replicas are similar and control is decentralized.
- To hide replication from the clients, replication usually entails very expensive forms of consistency
- A more scalable and common use for clouds may consist of stateless servers that do not try to coordinate.

Related Work

- GlassFish and JBoss support a “cluster mode”, where a load balancer may distribute requests among the available machines.
- Depending on the applications, consistency may be pushed back to the (non-replicated) database level, or may be entirely avoided. In such settings, the role of the cloud may be to dynamically grow or shrink the server farm, based on demand.
- Consistency required, keeping a large fraction of operational machines (virtual or physical), possibly a majority, is usually a very important problem.

Model

Faults:

- Crash-stop;
- Byzantine faults.

Malicious users have limited power due:

- Money needed to hire services;
- Inability to pick right targets;
- Computational resources necessary for a successful attack.

Attackers are unable to determine all the cloud provider resources and perform selective attacks to the running services.

Model (cont.)

Cloud has ***m*** PMs (Physical Machine). Each may run one or more of the ***v*** processes that provide some service and each process runs on its own VM (Virtual Machine) that provide some service.

To clarify the model is also considered:

- Urns and balls problem:
  - The urn is a PM;
  - The VMs are the balls.

VM failures correspond to drawing a ball from an urn.

In some cases, the attacker will put the ball again in the same urn, sometimes, will simply remove the ball.

Terms: ***without removal*** and ***removal***.

Theoretical Analysis

Independent VM failures

- The failure of a VM does not affect any other co-located VM.
- This may fit a crash-stop fault.
- The PM needs to have at least one operational VM, which is to say that the PM becomes unusable as soon as the last VM fails.
- Given the failure of *n ≤ v VMs*, can calculate the probability that a PM with vi VMs fails. We compute this value in Equation 1, for *vi* < *n* ≤ *v*, using the Hypergeometric distribution.
- *Pfi (n)* is the probability that *PM i* fails, after all its *vi VMs* have failed.

$$P_{f_i}(n) = \frac{\binom{v_i}{v_i}\binom{v-v_i}{n-v_i}}{\binom{v}{n}} = \frac{n \times (n-1) \times \ldots (n-v_i+1)}{v \times (v-1) \times \ldots (v-v_i+1)}$$

Theoretical Analysis (Cont)

VM failure contaminates other VMs

- Consider Byzantine attacks, where a malicious user may take over all the co-located VMs.
- In mathematical terms, we can treat this problem as an urns and balls problem, and use known results.
    - Urns represent a PM.
    - Balls represent the VM’s or processes.

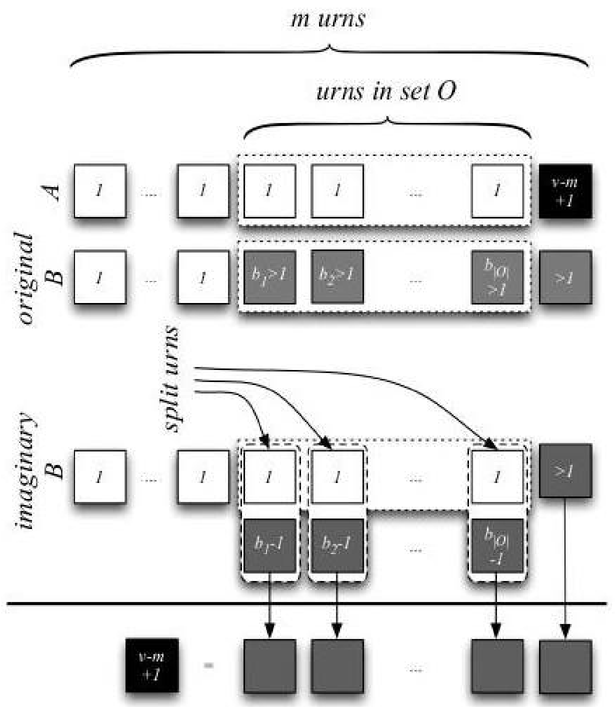
To distinguish between *correct* and *compromised VMs* we may assign colors to balls:

- White balls represent correct machines.
- Black balls represent compromised machines.

Theoretical Analysis (Cont)

Theorem 1.

Assume that we have ***v white balls*** distributed by 1 < m < v urns, such that each urn has at least one ball. Assume that the attacker works in successive turns, picking one ball at a time from a urn, and always putting back a black ball in the same urn.
The attacker does not select the ball or the urn. ***∀ k ∈ {1, . . . , m}***, ***P (X ≥ k)*** is maximized when ***m − 1*** urns have 1 ball, and the remaining urn has the remaining ***v − m + 1*** balls.



Theoretical Analysis (Cont)

Keeping a Majority of Virtual Machines

- The next question consider is the impact of machine failures on the number of VMs (or processes) that stay alive in a correct state.
- In many cases, this number might be more important than the number of different PMs where the replicas run.
- Concentrating many processes in the same (or few) machine(s) could reduce the average number of processes that survive (other) PM crashes.

  Interestingly, the answer is no.
- If we assume that all PMs have the same characteristics, at any given time t, the average number of VMs running is the same, independently of their distribution by the same set of PMs.

Theoretical Analysis (Cont)

If processes themselves do not fail, we have:

$$E[Z(t)] = \sum_{i=1}^{m} v_i \cdot P[Y_i(t) = 1] = P[Y(t) = 1]v$$

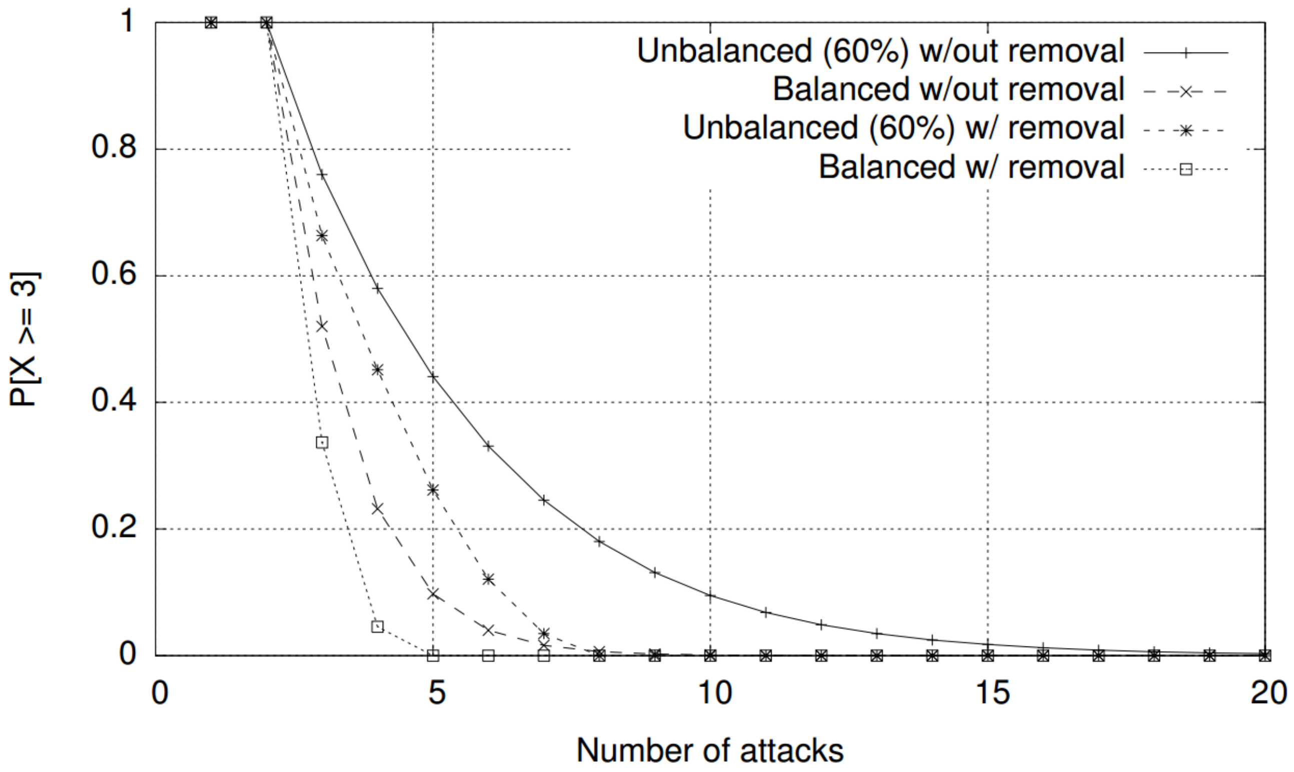
where Z(t) is a random variable that represents the number of VMs that are running at time t. Variable *vi* is the number of VMs running in machine *i*, Yi (t) is a random variable that assumes the value 0 if machine *i* is off at time *t* or 1 if the machine is on. Since we assume that this probability is the same for all machines, we can remove the subscript and simply write *P [Y (t) = 1]*.

Experimental Evaluation

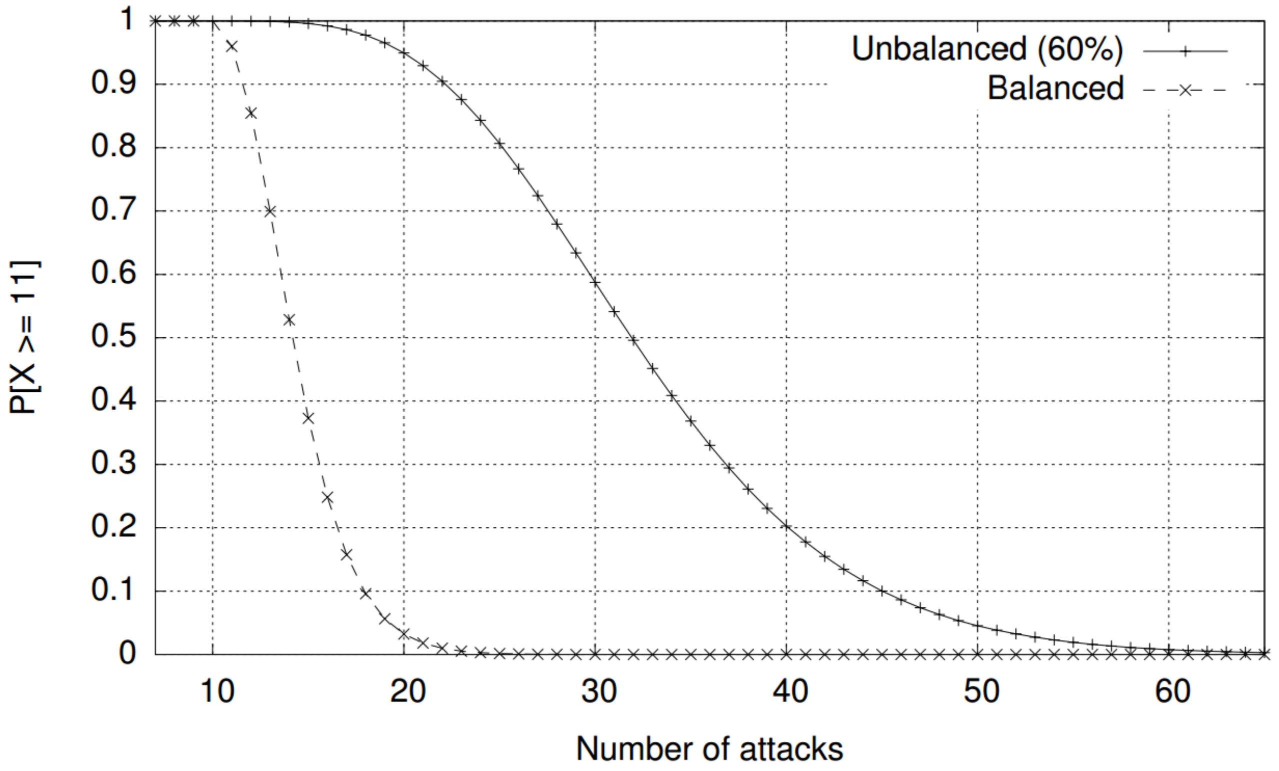
4 cases for `m = 5` PMs and `v = 10` VMs:

1. All the machines have same probability of receiving an attack ***(Balanced without removal)***;
2. A single machine has 60% probability, while the remaining machines has 40% chance of receiving an attack ***(Unbalanced (60%) without removal)***;
3. ***Balanced removal;***
4. ***Unbalanced (60%) removal.***

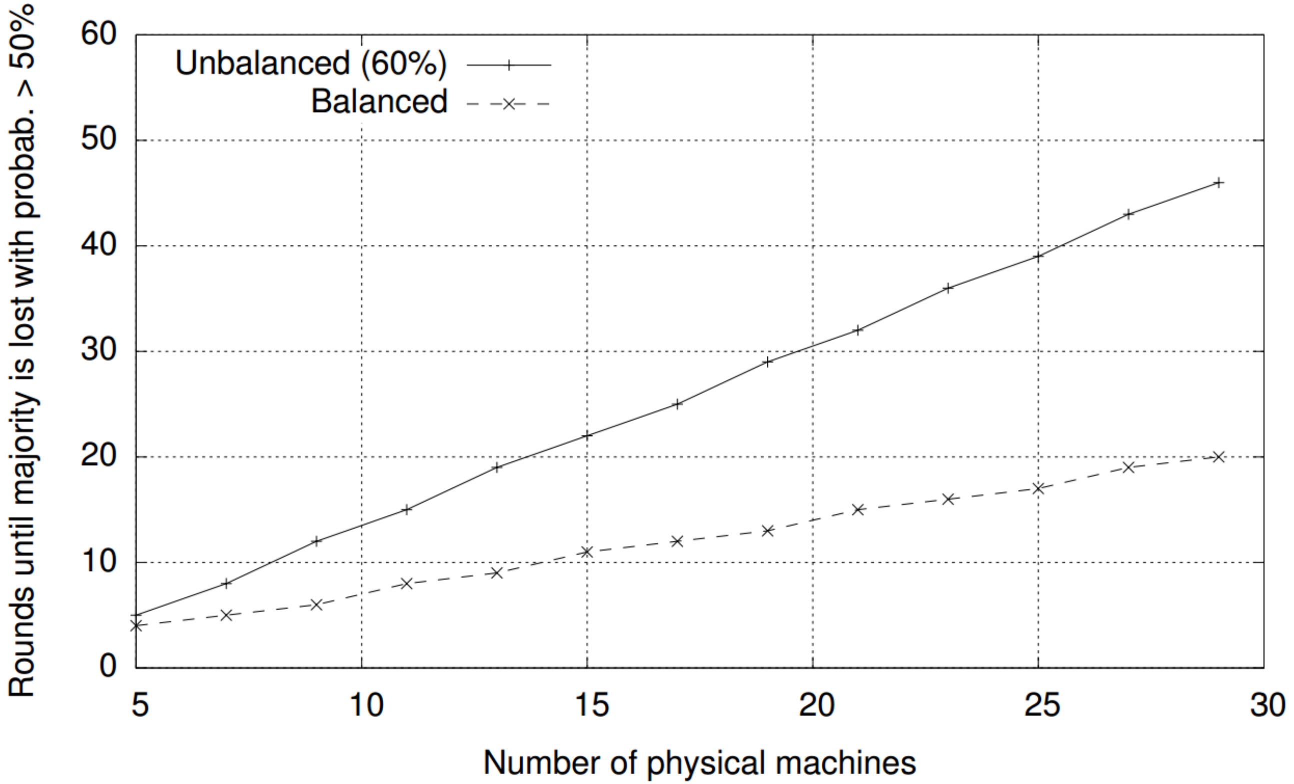
Experimental Evaluation (cont.)



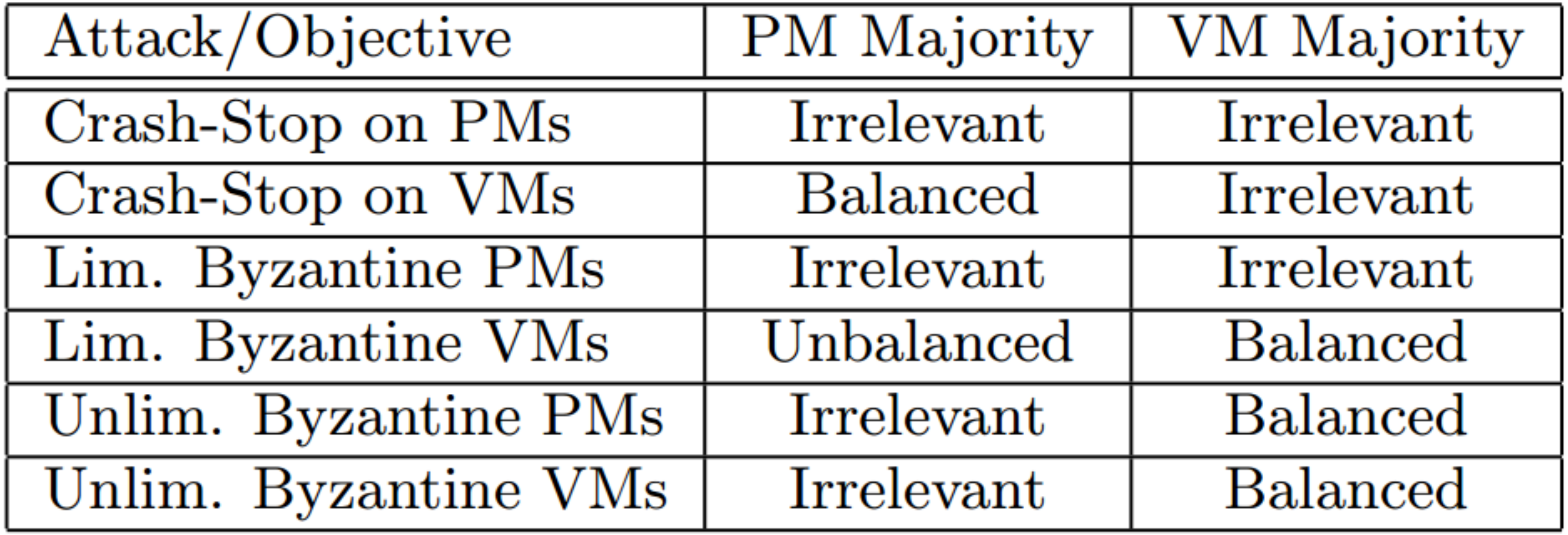
Experimental Evaluation (cont.)



Experimental Evaluation (cont.)



Placement Strategies of the Defense



Attack/Objective	PM Majority	VM Majority
Crash-Stop on PMs	Irrelevant	Irrelevant
Crash-Stop on VMs	Balanced	Irrelevant
Lim. Byzantine PMs	Irrelevant	Irrelevant
Lim. Byzantine VMs	Unbalanced	Balanced
Unlim. Byzantine PMs	Irrelevant	Balanced
Unlim. Byzantine VMs	Irrelevant	Balanced

Conclusion

Clouds are changing the surface of information technologies. However, concentration of resources in the same region, network, or even machine poses a challenge to designers of dependable systems.

The perspective of the cloud provider that needs to distribute processes or VMs by a given fixed set of PMs. Although intuition could suggest that evenly balancing the VMs would make a more dependable system, for many scenarios this is not the case: the distribution is either irrelevant or should be extremely unbalanced.

The user may anticipate himself to the cloud provider by distrusting it. In this case, the user might be tempted to hire the services of multiple providers to balance the VMs or the processes at his own will.

Questions??